

In re Application of GRIER et al.
Serial No. 09/842,270

REMARKS

The Office action has been carefully considered. The Office action rejected claims 15, 31-41, and 49 under U.S.C. §101 as being directed to non-statutory subject matter. Further, the Office action rejected claims 1-17, 19-22, 25-31, 42-43, 45-47, and 49 under U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,974,470 to Hammond et al. ("Hammond"). Further yet, the Office action rejected claims 32-41 under U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,185,734 to Saboff et al. ("Saboff"). Still further, the Office action rejected claims 18, 24, and 44 under U.S.C. §103(a) as being unpatentable over Hammond. Finally, the Office action rejected claims 23 and 48 under U.S.C. §103(a) as being unpatentable over Hammond in view of Saboff.

In addition to the claim rejections noted above, the Office action has provisionally rejected claims 1, 3-5, 7-22, 24-26, 31, 42, 43, and 45-49 under the judicially created doctrine of obviousness-type double patenting as being obvious in view of claims 1-4, 12, 16-22, 26, and 27 of U.S. Patent No. 6,871,344. Although applicants do not agree with these rejections, and submit that the present claims should in no way be limited by any interpretation of any claims of U.S. Patent No. 6,871,344, applicants file herewith a terminal disclaimer thereby eliminating the obviousness-type double patenting rejection.

The Office action also objected to the specification over an incorrect reference in the claim of priority. Applicants have corrected the error in the specification which eliminates any erroneous reference to an incorrect copending application, thus, overcoming the objection.

In re Application of GRIER et al.
Serial No. 09/842,270

By present amendment, claims 1, 16, 32, 39, 41, and 42 have been amended for clarification and not in view of the prior art. Applicants submit that the claims as filed were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims and not for reasons related to patentability. Reconsideration is respectfully requested.

Applicants thank the Examiner for the interview held (by telephone) on April 18, 2006. During the interview, the Examiner and applicants' attorney discussed the claims with respect to the prior art. The essence of applicants' position is incorporated in the remarks below.

Prior to discussing reasons why applicants believe that the claims in this application are clearly allowable in view of the teachings of the cited and applied references, a brief description of the present invention is presented.

The present invention is directed to a system and method for a computing infrastructure that allows applications to run with specified versions of dependent assemblies, wherein each assembly may exist and run side-by-side on the system with other versions of the same assembly being used by other applications. More specifically, when executed, an application may build an activation context for distinguishing between actual versions of various assemblies. The activation context may provide a manifest to specify any desired assembly versions on which it is dependent which includes actual references to the particular version of each needed assembly. Similarly, each assembly may have an assembly manifest that specifies the versions of assemblies on which it is dependent. The particular

In re Application of GRIER et al.
Serial No. 09/842,270

versions of the assemblies used by the applications may be tracked in a separate and distinct file such that if assemblies change as updates are implemented, the application does not also need to be changed and updated, rather just the manifest. In this manner, the application that works better with (or requires) a particular version of an assembly may retrieve the particular version of the assembly while other applications that are not as dependent on a very specific version may use the updated version of assemblies as they become available.

For example, during an initialization phase, *i.e.*, when an application is first launched, an activation context may be created for the application based on the manifests to map version independent names to a particular assembly version maintained on the system. While the application is in a running phase, for any globally named object that the application wants created, the activation context may be accessed to identify the application's or assembly's manifest-specified version. The manifests and activation context constructed therefrom, thus, may isolate an application from assembly version changes.

Note that the above description is for example and informational purposes only, and should not be used to interpret the claims, which are discussed below.

§101 Claim Rejections

The Office action rejected claims 15, 31-41, and 49 as being directed to non-statutory subject matter because these claims are recited as computer-readable media that may embody modulated data signals and/or carrier waves. The Office action, by acting upon *Interim Guidelines for Examination of Patent*

In re Application of GRIER et al.
Serial No. 09/842,270

Applications for Patent Subject Matter Eligibility (1300 OG 142), Annex IV, has then concluded that these claims are directed to non-statutory subject matter by introducing one definition among many from the specification. Applicants respectfully disagree and submit that claims 15, 31-41, and 49 are directed toward a computer-readable medium well within defined statutory subject matter.

MPEP § 2106(IV)(B)(1a) specifically states that "a claimed computer readable medium encoded with a data structure *defines* (emphasis added) structural and functional interrelationships between the data structure and the medium which permit the data structure's functionality to be realized, and is thus statutory." In contrast, MPEP § 2106(IV)(B)(1a) goes on to state further "[d]ata structures *not claimed as embodied in computer-readable media* (emphasis added) are descriptive material per se and are not statutory because they are not capable of causing functional change in the computer." By organizing a data structure with the claimed sets of data on a computer readable medium, structural and functional interrelationships between the data structure and the medium are realized and defined as statutory *per se*. Applicants maintain that claims 15, 31-41, and 49, as originally entered recite statutory subject matter.

Notwithstanding, applicants have nevertheless amended claims 15, 31, 32, 39, 41, and 49 to recite additional sufficient functionality, *i.e.*, a computer-readable storage medium, so as to remove any doubt whatsoever about the statutory nature of the recitations. Applicants respectfully submit that claims 15, 31-41, and 49 are directed to statutory subject matter and no further correction is required.

In re Application of GRIER et al.
Serial No. 09/842,270

\$102 Claim Rejections citing Hammond

Turning to the first independent claim, amended claim 1 essentially recites a computer-implemented method, comprising receiving a request from executable code to load an assembly, the request not including assembly version data, including when the assembly is among a plurality of assemblies having at least some components located in a same directory, building an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with the executable code in response to the request to load an assembly, consulting information in a manifest associated with the executable code and stored separate from the executable code to determine a particular version of the assembly in response to the building of the activation context, and providing the particular version of the assembly for use by the executable code.

The Office action rejected claim 1 as being anticipated by Hammond. More specifically, the Office action contends that Hammond teaches receiving a request from executable code to load an assembly, the request not including assembly version data. Column 5, line 58 to column 6, line 12 of Hammond is referenced. Further, the Office action contends that Hammond teaches building an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with the executable code in response to the request to load an assembly. Column 8, lines 23-58 and column 7, lines 59-64 of Hammond is referenced. Further, the Office action contends that Hammond

**In re Application of GRIER et al.
Serial No. 09/842,270**

teaches consulting information associated with the executable code to determine a particular version of the assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Finally, the Office action contends that Hammond teaches providing the particular version of the assembly for use by the executable code. Column 5, lines 27-30 of Hammond is referenced. Applicants respectfully disagree.

As has been presented in previous Office action response, Hammond is directed, generally, to a software method of providing a patch to an operating system that modifies the Application Program Interface (API) call logic. When a program is executed, any called modules are then called according to the new call logic instead of the original call logic of the operating system. See Hammond's abstract, generally. The software patch provides the user with the option of setting "location rules" that govern what path that the call logic will use to locate the called modules. See column 5, lines 34-40 of Hammond.

One such location rule (specifically cited by the Office action) that may be set is the "version rule." When set, the version rule bit is an indication to the call logic that only one specific module must be called in conjunction with the application. However, "version" is simply a name given to the rule to distinguish between calling any module and calling a specific module. The actual version, however, is not specified by the system disclosed in Hammond. Rather, when the version rule is set, the call logic will only look to a single location to find the module, but the call logic remains unaware of the actual version of the module being called. See column 7, lines 51-67 of Hammond. This is further bolstered by the fact that

In re Application of GRIER et al.
Serial No. 09/842,270

Hammond leaves unaddressed any scenario in which two versions of the same module may be located in the same directory. In fact, the solution in Hammond is to necessarily avoid that very scenario by requiring that different versions of the same module be stored in separate directories so that the single specified location indicates the unique location of the specified module when the "version rule" (which is really a location indicator) is set.

For example, Hammond refers to a scenario wherein two DLL files are referred to as version 1.0 and version 2.0 but the call logic is unaware of the DLL file being called. The version rule merely directs the call logic to a location which allows the call logic to load whatever DLL file happens to be in the directory, *regardless of what it may be, (i.e., it will load version 2.0 because it is located at a specified location and not because it is the second version of the DLL).* If version 1.0 and version 2.0 were located in the same directory (which is advantageous for ease of storage reasons and consistency), the system of Hammond would crash or potentially load the wrong version because it cannot distinguish which module to load since two exist in the directory specified by the location pointer. Thus, despite the confusing choice of Hammond to refer to the location-determining bit as the "version rule," Hammond does not teach distinguishing between actual versions of DLL files or any other files, modules or assemblies based on version information.

The Office action further cites a version key taught by Hammond as also distinguishing between different versions of assemblies. In Hammond, however, the version key (unfortunately inappropriately named relative to the term "version" as is normally understood) does not distinguish between actual versions of

In re Application of GRIER et al.
Serial No. 09/842,270

assemblies as is well-understood in the industry. A version number is indicative of a point-in-time sequential build, e.g., version 2.0 is newer and intended to be better than version 1.0. In Hammond, the version key merely computes sums for files based on the number of segments, segment size, number of resources, and the like. See column 7, lines 31-50 of Hammond. A mathematical sum representing number of segments, segment size, and number of resources is not the same as a version number indicative of a point-in-time sequential build. Thus, the version key of Hammond only contains a mathematical sum for representing files, which applications are then able to associate with a file based on this mathematical sum (which is typically so intricately produced and large that it is, in effect, unique for each file). Using this kind of system distinguishes between files based solely upon the order in which bits appear in the file. This cannot possibly be construed to distinguish on anything in the file other than its essentially random bit structure and certainly cannot distinguish based on version numbers of the file.

Notwithstanding these differences, claim 1 has been amended to recite receiving a request from executable code to load an assembly, the request not including assembly version data, including when the assembly is among a plurality of assemblies having at least some components located in a same directory. Maintaining, distinguishing, and determining the actual version of an assembly among a plurality of assemblies in the same directory, is unquestionably not the same as determining the location of a DLL file that happens to be named by a version number as is taught by Hammond. It is also different from maintaining

In re Application of GRIER et al.
Serial No. 09/842,270

mathematical sums about files and simply labeling the sum list as a version key as is also taught by Hammond.

Further yet, the Office action has interpreted an "activation context" to be a broad term that would encompass any environment or "context" in which an application is initiated. Applicants maintain that this is an overly broad and incorrect interpretation of the term. Understanding that the Office action must necessarily consider the broadest possible reasonable interpretation of the claims, the claims are not meant to be read in a vacuum. The specification of the present invention describes an example activation context in terms of an XML-based document. In general, the initialization mechanism constructs a dependency graph or other structure that comprises a full set of assemblies that an application and its dependent assemblies, and their dependent assemblies, and so on, will need, and with the completed dependency graph, constructs the activation context. Typically, there is an activation context for each application that has an expressed assembly dependency, and each activation context includes a mapping table. Certainly then, Hammond cannot be construed to teach these recitations of claim 1 as Hammond does not teach the concept of building an activation context in response to a request to load an assembly in this sense.

Therefore, the teachings of Hammond do not disclose the method recited in claim 1. For at least the foregoing reasons, applicants submit that claim 1 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 2-15, by similar analysis, are allowable. Each of these claims depends either directly or indirectly

In re Application of GRIER et al.
Serial No. 09/842,270

from claim 1 and consequently includes the recitations of independent claim 1. As discussed above, Hammond fails to disclose the recitations of claim 1 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 1 noted above, each of these dependent claims includes additional patentable elements.

For example, claim 5 essentially recites that consulting information associated with the executable code to determine a particular version of the assembly includes searching for a mapping from a version independent name provided by the executable code to a version specific assembly. As discussed above, Hammond does not teach consulting information associated with the executable code to determine a particular version of the assembly. Instead, Hammond teaches determining the *location* of a DLL file specified to be loaded and cannot distinguish between versions. Therefore, Hammond cannot possibly teach searching for a mapping from a version-independent name provided by the executable code to a version-specific assembly. Although, Hammond discloses a "version map," this concept in Hammond is merely an identification of aliases used when multiple calls of identically named modules are made. Hammond provides a means for renaming modules that happen to be named by the same file name (library.dll, for example) that are called more than once and a means for keeping track of the resulting names in what Hammond has chosen to call "a version map." The version map of Hammond, which is not actually aware of versioning information of its stored files, cannot be construed to read on identifying specific versions of an assembly that are mapped to version independent names of

**In re Application of GRIER et al.
Serial No. 09/842,270**

assemblies as recited in claim 5. Applicants submit that claim 5 is allowable for at least this additional reason.

Turning to the next independent claim, amended claim 16 essentially recites a computer-implemented method, comprising building an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with executable code, the activation context identifying dependency information, interpreting the dependency information associated with the executable code, the dependency information identifying at least one particular version of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory, and associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information.

The Office action rejected claim 16 as being anticipated by Hammond. More specifically, the Office action contends that Hammond teaches interpreting dependency information associated with executable code, the dependency information identifying at least one particular version of an assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Further, the Office action contends that Hammond teaches building an activation context, the activation context identifying dependency information. Column 8, lines 23-58 of Hammond is referenced. Further, the Office action contends that Hammond teaches associating with the executable code at least one mapping based on the dependency

In re Application of GRIER et al.
Serial No. 09/842,270

information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information. Again, Column 7, line 51 to column 8, line 5 of Hammond is referenced. Applicants respectfully disagree.

As discussed above, the method and system disclosed in Hammond does not teach utilizing information associated with the application itself to determine which actual version of a module is required during execution. Rather, Hammond teaches a system and method for providing location rules that assist the operating system in determining the location of modules to load when applications are executed. Hammond does not teach interpreting dependency information associated with executable code, the dependency information identifying at least one particular version of an assembly, as recited in claim 16. At best, Hammond teaches interpreting location rules to determine the location of the module to call.

Furthermore, since Hammond only teaches determining the location of modules to call, Hammond cannot possibly be construed to teach associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information as recited in claim 16. Again, at best, Hammond teaches mapping different alias names assigned to multiples calls of the same version of a single module.

In contrast, claim 16 recites building an activation context that distinguishes between versions of assemblies based on the actual version associated with the

In re Application of GRIER et al.
Serial No. 09/842,270

executable code in response to a request to load an assembly. The method of the present invention may consult information stored in an activation context to determine which actual version of an assembly may be needed by the executable code. In this manner, the actual version of the assembly needed may be stored in a document which may be an XML document. Maintaining, distinguishing, and determining the actual version of an assembly, as is generally the case with the recitations of claim 16, is not the same as determining the location of a DLL file that happens to be named by a version number as is taught by Hammond. It is also different from maintaining mathematical sums about files and simply labeling the sum list as a version key as is also taught by Hammond.

Notwithstanding these differences, claim 1 has been amended to recite the dependency information identifying at least one particular version of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory. Maintaining, distinguishing, and determining the actual version of an assembly among a plurality of assemblies in the same directory is clearly not the same as determining the location of a DLL file that happens to be named by a version number as is taught by Hammond. It is also different from maintaining mathematical sums about files and simply labeling the sum list as a version key as is also taught by Hammond.

For at least the foregoing reasons, applicants submit that claim 16 is allowable over the prior art of record for at least these reasons.

Applicants respectfully submit that dependent claims 17, 19-22, and 25-31 by similar analysis are allowable. Each of these claims depends either directly or

In re Application of GRIER et al.
Serial No. 09/842,270

indirectly from claim 16 and consequently includes the recitations of independent claim 16. As discussed above, Hammond fails to disclose the recitations of claim 16 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 16 noted above, each of these dependent claims includes additional patentable elements.

Turning to the next independent claim alleged as anticipated by Hammond, amended claim 42 generally recites an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends, each assembly version corresponding to an assembly having version information associated therewith and contained in a directory structure among a plurality of assemblies, an activation context, the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data, the activation context relating at least one version independent assembly identifier provided by the executable code to a version specific assembly, and a version matching mechanism configured to access the activation context that distinguishes between versions of assemblies based on the actual version to relate a version independent request from the executable code to a version specific assembly.

The Office action specifically contends that Hammond teaches an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends. Column 7, line 51 to column 8,

**In re Application of GRIER et al.
Serial No. 09/842,270**

line 5 of Hammond is referenced. The Office action also contends that Hammond teaches an activation context, the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data, the activation context relating at least one version independent assembly identifier to a version specific assembly. Column 8, lines 23-58 of Hammond is referenced. Finally, the Office action contends that Hammond teaches a version matching mechanism configured to access the activation context to relate a version independent request from the executable code to a version specific assembly. Column 7, line 51 to column 8, line 5 of Hammond is referenced. Applicants respectfully disagree.

As discussed above, no where in Hammond can there be found any reasonable teaching of an activation context as used herein and consistent with the specification. For that matter, Hammond does not show any cognizance of an activation context in this sense, let alone the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data as recited in claim 42. At best, Hammond teaches a version map for mapping different alias names assigned to multiples calls of the identically named modules.

Notwithstanding this deficiency, claim 42 has been amended to point out that the assembly version on which the executable code depends may be of an assembly among a plurality of assemblies in a directory structure. As discussed above, Hammond cannot handle a different scenario wherein versions of the same module are located in the same directory structure. Thus, Hammond cannot

In re Application of GRIER et al.
Serial No. 09/842,270

possibly be construed to teach a system for dealing with this issue. For at least the foregoing reasons, applicants submit that claim 42 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 43, 45-47, and 49 by similar analysis are allowable. Each of these claims depends either directly or indirectly from claim 42 and consequently includes the recitations of independent claim 42. As discussed above, Hammond fails to disclose the recitations of claim 42 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 42 noted above, each of these dependent claims includes additional patentable elements.

§102 Claim Rejections citing Saboff

Amended claim 32 recites a computer-readable storage medium having stored thereon a data structure, comprising a first data store operable to store a first set of data comprising a name of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory, a second data store operable to store a second set of data comprising a version of the assembly, a third data store operable to store a third set of data comprising at least one item of the assembly, and a fourth data store operable to store a fourth set of data comprising binding path data to each item in the third set of data, wherein each data store is operable to provide information to an activation context that distinguishes between versions of assemblies based on the actual version when executable code is executed.

In re Application of GRIER et al.
Serial No. 09/842,270

The Office action rejected claim 32 as being anticipated by Saboff. More particularly, the Office action contends that Saboff teaches a first data store operable to store a first set of data comprising a name of an assembly. Service 509 in FIG. 6 and column 9, lines 11-18 of Saboff are referenced. Further, the Office action contends that Saboff teaches a second data store operable to store a second set of data comprising a version of the assembly. Service 513 in FIG. 6 and column 9, lines 53-55 of Saboff are referenced. Still further, the Office action contends that Saboff teaches a third data store operable to store a third set of data comprising at least one item of the assembly. State type 510 and 511 of FIG. 6 and column 9, lines 19-39 of Saboff are referenced. Finally, the Office action contends that Saboff teaches a fourth data store operable to store a fourth set of data comprising binding path data to each item in the third set of data. Path 507 of FIG. 6 and column 9, lines 5-7 of Saboff are referenced. Applicants respectfully disagree.

As has been presented in previous Office action responses, Saboff is directed, generally, to a registry for maintaining versions of services such that multiple applications may use the various versions of the same services. Accordingly, Saboff at best discloses a data structure having a number of fields that contain values for use in its service maintenance system. In particular, Saboff discloses a data structure having a name of a service (509), a version of the service (513), a type or state of the service (510 and 511), and a path to where the service is located (507). Nowhere in Saboff, however, is there any disclosure of a

In re Application of GRIER et al.
Serial No. 09/842,270

system that may deal with more than one version of an assembly that may be located in the same directory.

In contrast, claim 32 recites a first set of data comprising an assembly. As argued in the previous Office action response, an assembly is not a service as in Saboff. Further, claim 32 recites a third set of data comprising at least one item of the assembly. Again, an assembly is not a service and one item in the assembly is quite different from the type or state of a service. Thus, the contention by the Office action that the disclosures of Saboff correspond on a one-for-one basis (as required by law to support a § 102 rejection) to the recitations of claim 32 is not supported by Saboff.

Furthermore, Saboff cannot possibly teach the recitations of claim 32 because of the nature in which the fields of the data structure differ. Saboff merely teaches the use of path data to indicate the location of the service identified in the service field. Quite differently, claim 32 recites a fourth set of data comprising binding path data to each item in the third set of data. That is, the binding path does not represent a location of the assembly itself, but rather a relationship between each item in the third set of data and an outside source, such as an API or call program. Not only does the data in the fourth set correspond to a different set of data than what is disclosed in Saboff (item in an assembly as opposed to a service), it also corresponds in a different manner (binding path as opposed to mere location). A binding path necessarily implies a relationship between two items whereas a location merely indicates information about one item. Further yet, claim 32 recites an activation context that distinguishes between versions of

In re Application of GRIER et al.
Serial No. 09/842,270

assemblies based on actual versions. Saboff simply does not teach, in any way, an activation context in this sense.

Notwithstanding these differences, claim 32 has been amended to emphasize another fundamental difference, namely that no prior art of record teaches a system or data structure that allows for distinguishing between different versions of the same assemblies that are located in the same directory. Applicants submit that claim 32 is allowable over the prior art for at least the foregoing reasons.

Applicants respectfully submit that dependent claims 32-38, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 32 and consequently includes the recitations of independent claim 32. As discussed above, Saboff fails to disclose the recitations of claim 32 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 32 noted above, each of these dependent claims includes additional patentable elements.

Regarding independent claims 39 and 41 as well as dependent claim 40, these claims include recitations generally directed to a computer-readable medium encoded with a data structure having a set of data generally relating to an assembly. As discussed above regarding the recitation of claims 32-38, an assembly is not a service as described in Saboff, and, consequently by similar analysis, Saboff also fails to disclose the recitations of claims 39-41. Furthermore, claims 39-41 are allowable for their additional patentable elements including amended recitations therein directed to the capability of storing information in the

In re Application of GRIER et al.
Serial No. 09/842,270

same directory structure. Applicants submit that claims 39-41 are allowable over the prior art of record for at least these reasons.

§103(a) Claim Rejections

The Office action rejected claims 18, 23, 24, 44, and 48 as being unpatentable either over Hammond alone or in view of Saboff. Claims 18 and 23-24 depend from claim 16 and claims 44 and 48 depend from claim 42. As shown above, Hammond, whether considered alone or in any permissible combination at law with any prior art of record, including Saboff, does not teach or suggest the recitations of claims 16 and 42 respectively for at least the reasons discussed above with respect to the §102 rejections of these independent claims. Thus, applicants submit that each of these dependent claims are also allowable for at least the reasons that each respective independent claim is allowable as was discussed above. More particularly, applicants submit that claims 18, 23, and 24 are allowable for at least the reasons that claim 16, the claim from which these claims depend, is allowable. Additionally, applicants submit that claims 44 and 48 are allowable for at least the reasons that claim 42, the claim from which these claims depend, is allowable.

For example, neither Hammond nor any other prior art of record, whether considered alone or in any permissible combination at law, discloses or suggests the recitation of an application manifest associated with the executable code by being stored in a common folder with an application executable file that corresponds to the executable code as recited in claim 18. Nor does Hammond, or

In re Application of GRIER et al.
Serial No. 09/842,270

any other prior art of record, whether considered alone or in any permissible combination at law, disclose or suggest the recitations of an initialization mechanism adding information that corresponds to assembly dependency data to an activation context as recited in claim 48.

Further, by law, in order to modify a reference to reject claimed subject matter, there must be some teaching or suggestion outside of applicants' teachings to do so. Neither Hammond nor Saboff have any such teachings or suggestions as to any such modification, let alone any teaching or suggestion as to how either Hammond's system or Saboff's system could be modified, or why it might be desirable to do so. The only other way in which Hammond or Saboff could be modified or combined to reach applicants' claimed invention (assuming for the sake of argument that such a modification is even possible) is via applicants' own teachings, which is impermissible by law.

For at least the reasons discussed above with respect to both the §102 and §103 rejections, applicants submit that all the claims are patentable over the prior art of record. Reconsideration and withdrawal of the rejections in the Office action is respectfully requested and timely allowance of this application is earnestly solicited.

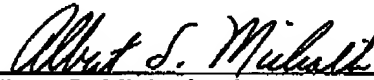
In re Application of GRIER et al.
Serial No. 09/842,270

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-49 are patentable over the prior art of record, and that the application is in good and proper form for allowance. A favorable action on the part of the Examiner is earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,



Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
/w
(425) 836-8957 (facsimile)

In re Application of GRIER et al.
Serial No. 09/842,270

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that this Response, along with transmittal, RCE, credit card payment form, and facsimile cover sheet, are being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. 1.6(d) on the date shown below:

Date: June 12, 2006


Albert S. Michalik

2501 Fourth Amendment